13. ročník
Žilina 1. – 3. 7. 2025
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.fri.uniza.sk
OSS Conf

OSSConf 2025: 1–8

O jedné mapě na zakázku, část 1: Teorém 4 barev prakticky

PAVEL STŘÍŽ (CZ)

Abstrakt. Tento článek je úvodem do sazby map pomocí R a TEXu. Užíváme data z OpenStreetMap. V této první části z několika článků se zaměřujeme na užití teorému čtyř barev na obarvení regionů. Jádro článku bylo užití jednoho z řešitelů (SAT, MIP, CP, SMT) a jejich stručné srovnání. Použili jsme Python, který generuje .pi soubor pro program Picat. Ten má výhodu v tom, že na jednom řádku se lze mezi řešiteli přepínat.

Klíčová slova. Optimalizace, R, Python, Picat, OpenStreetMap.

ABOUT PREPARATION OF ONE MAP, PART 1: THE FOUR COLOUR THEOREM IN PRACTICE

Abstract. The article is an introduction to typesetting maps using R and TEX. We are using data from OpenStreetMap. In this first part of several articles, we focused on applying The Four Colour Theorem to fill the regions. The main approach was to use optimization techniques (SAT, MIP, CP, SMT) and briefly compare them. We used Python to generate .pi code for the Picat software. It has feature that we can switch between mentioned solvers editing one line of code.

Keywords. Optimization, R, Python, Picat, OpenStreetMap.

1. Rešerše: The Four Colour Theorem

Během mých výletů ve světě wordcloudů jsem otevřel starší problém, a to vysázet mapu ČR a SR s erby měst. V pozadí jde o to, že místo bodů na spirále mohu využít jediný bod (GPS souřadnici), tedy by to mělo být výrazně rychlé i při tisícovkách měst. Aleš Kozubík mi poradil, ať zkusím něco (kraje, okresy, města) v pozadí vybarvit přes teorém 4 barev (zkracováno na 4CT). Tento teorém je mezi informatiky a matematiky slavný, protože byl první, který byl dokázán za asistence počítačů. V sedmdesátých letech 20. století kontroverzní téma, nyní již záležitost běžně přijímána.

Ony významné publikace jsou [3,4]. Čtenářům bych doporučil článek Georgese Gonthiera [2], který poutavě píše o historii teorému, ale také o moderní asistenci, speciálně přes program Coq, nyní nazýván Rocq, `https://en.wikipedia.org/wiki/Rocq`. Z novějších knih zmíním *A Guide to Graph Colouring* od R. M. R. Lewise [1]. Poměrně podrobně se zabývá heuristickými algoritmy.

Populární formou o důkazu mluví na kanálu Numberphile, `https://www.youtube.com/watch?v=NgbK43jB4rQ`.

Jak v mé přítomnosti vyslovíte teorii grafů, ihned se mi vybaví nástroj NetworkX. Pokud však nahlédneme do `https://networkx.org/documentation/stable/reference/algorithms/coloring.html`, zjistíme, že se minimálního počtu barev nedočkáme. To jsem zbystřil. Na YouTube chlapík, `https://www.youtube.com/watch?v=AyrrZ4PCyws`, šel také do rekurze.

Optimální strategii nenalezneme ani v nástroji `gcol` v Pythonu. `https://rhydlewis.eu/gcol/` To je vedlejší produkt ke knize [1]. To už mi bylo divné.

Pak už to přece jenom začalo být zajímavější. Prvně jsem to klučinu viděl řešit přes knihovnu `simpleai`, `https://pypi.org/project/simpleai/`, potažmo `pydot`, `flask` a `Flask-SQLAlchemy`, viz `https://www.youtube.com/watch?v=hV6qTfFSwCk`. Dokumentace hovoří o CSP algoritmech, a to už je člověk doma.

Přes Python jsem zahlédl Knuthův Algoritmus X, `https://stackoverflow.com/questions/42863543/applying-the-4-color-theorem-to-list-of-neighbor-polygons-stocked-in-a-graph-arr`. To jsem nezkoušel, ale zpozornil jsem. Přes mé oblíbené SAT řešitele jsem zahlédl tuto implementaci, `https://mathoverflow.net/questions/419808/is-there-any-fast-implementation-of-four-color-theorem-in-python`. Kdekoliv zahlédnu slavný program Gurobi, tak si představím svobodný program Symphony či Scip. To už jsem si říkal, to bude chtít něco zkusit vlastními silami. Pro tu srandu.

Pro seriózní badatele doporučuji `https://cran.r-project.org/web/views/Optimization.html`, já jsem zde bádání pozastavil.

Aktuální a fascinující ukázky představuje `https://four-color-theorem.com`, resp. `https://github.com/stefanutti/maps-coloring-python`. Ještě zmíním jeden tip: u map mě fascinovaly experimenty v R na `https://fronkonstin.com/`. Doporučuji cca 15 HTML stránek si prolistovat. Jsou to svěží grafické ukázky.

2. Vstup na pole R

Analýza dat se v těchto dnech nejčastěji bije na poli mezi Pythonem a R. Pro běžného smrtelníka je dobré umět obojí, byť pracovat s R je nezvyklé. Pro mne je stále nezvyk všude vidět `c()`, což je zápis dat do vektoru nebo seznamu (combine), `rbind` či `cbind`, což je rozhození dat do řádku či sloupce. Je to i po letech nezvyk.

V každém případě jsem narazil na blog z roku 2015, `https://www.r-bloggers.com/2015/11/solving-the-map-coloring-problem-in-r/`, a že to zkusím. Po chvíli hledání jsem našel autorův účet, viz `https://github.com/hunzikp/MapColoring`. Během instalace se však dozvíte, že CTAN nadále nepodporuje knihovny `rgeos` a `rgdal`, že je potřeba se obrátit na knihovnu `sf`. Knihovna vás pohltí, nemluvě o animaci hexaloga na `https://r-spatial.github.io/sf/`. Jak by řekl klasik: „Tohle je jiný svět!“

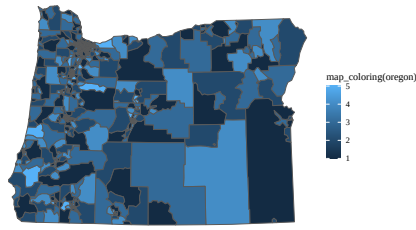
A to už začalo být zajímavé. Knihovna je podrobně popsána, a když nahlédneme na `https://r-spatial.github.io/sf/articles/sf1.html`, už to tam

body a polygony jen lítá. Bližší zkoumání mě dovezlo až ke knihovně `ggredist`, která už obarvení mapy umí. Ačkoliv si lze navolit snahu o minimum barev, autoři píší, že umí jen *greedy* funkci. Pokud se pohybujete na poli optimalizace, tak na tohle slovo jste alergičtí, protože bývá hned vedle *naïve* řešení. Berte to, prosím, s rezervou. Na vysvětlení principů však stav ideální.

V Linuxu jsem si pracovně odinstaloval knihovnu `libmariadb-dev` a doinstaloval `libudunits2-dev` a `libgdal-dev`. Pokud tak neučiníte, R vás bude informovat o neúspěšné instalaci. Potřebujete totiž vedle knihovny `ggredist` i `units`, `sf` a `ggplot2`.

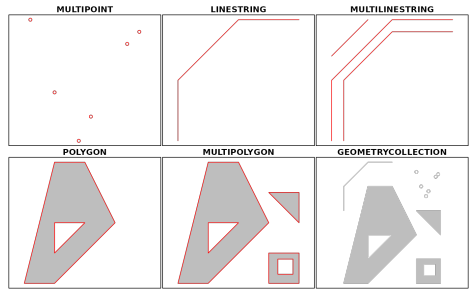
Pokud jsem si vzal základní ukázku a vztáhl na celý stát Oregon, dostáváme:

```
library(ggplot2)
library(ggredist)
data(oregon)
map_coloring(oregon)
ggplot(oregon, aes(fill = map_coloring(oregon))) + geom_sf() + theme_map()
```



3. Hlouběji do R

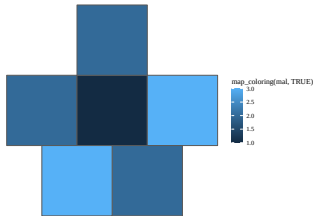
Dobře, základní ukázka je v pořádku, jak to však funguje? Struktura je taková, že z bodů vznikají úsečky, z těch polygony, či seznamy bodů, úseček či polygonů. Nejvyšší úroveň je kolekce geometrických útvarů. U seznamu polygonů je důležité, že na prvním místě je vnější křivka, další položky v seznamu jsou výřezy. Je to trochu nezvyk, ale u mapových podkladů to tak je.



Zkusíme ukázku. Připravíme si šest čtverců a necháme si je obarvit. Z bodů vzniká seznam, ze seznamu polygon, z polygonů kolekce pro knihovnu `sf`. Následně si necháme proměnnou vypsat, vykreslit přes `plot` a `ggplot`. V posledním bloku se dopočítají sousedské vztahy, vypíše a vykreslí se výsledek.

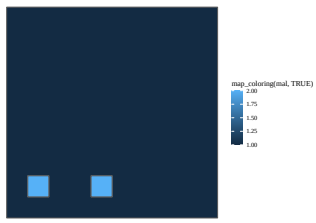
```
p1<-rbind(c(0,0), c(1,0), c(1,1), c(0,1), c(0,0))
```

```
p2<-rbind(c(1,0), c(2,0), c(2,1), c(1,1), c(1,0))
p3<-rbind(c(2,0), c(3,0), c(3,1), c(2,1), c(2,0))
p4<-rbind(c(1,1), c(2,1), c(2,2), c(1,2), c(1,1))
p5<-rbind(c(1.5,0), c(2.5,0), c(2.5,-1), c(1.5,-1), c(1.5,0))
p6<-rbind(c(0.5,0), c(1.5,0), c(1.5,-1), c(0.5,-1), c(0.5,0))

mpol1<-st_polygon(list(p1)); mpol2<-st_polygon(list(p2))
mpol3<-st_polygon(list(p3)); mpol4<-st_polygon(list(p4))
mpol5<-st_polygon(list(p5)); mpol6<-st_polygon(list(p6))

mal1<-st_sfc(list(mpol1,mpol2,mpol3,mpol4,mpol5,mpol6))
mal1
plot(mal1)
ggplot(mal1) + geom_sf() + theme_map()

mal<-st_sf(geom=mal1)
map_coloring(mal)
ggplot(mal, aes(fill = map_coloring(mal,TRUE))) + geom_sf() + theme_map()
```

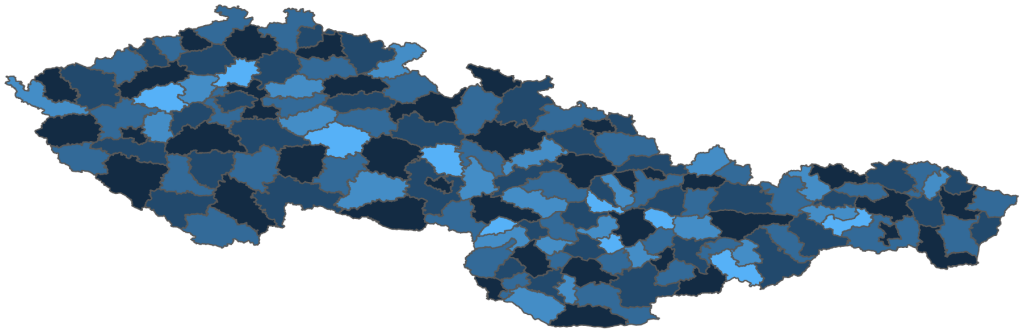


Abych ještě zprůhlednil tvorbu výřezů do polygonu, zde je jednoduchá ukázka. Do čtverce naděláme dvě čtvercové díry, resp. dva nezávislé polygony, které se mají také obarvit. Tato úvaha se nám bude dál hodit: Brno versus Brno-okolí, Košice versus Košice-okolí.

```
p10<-rbind(c(0,0), c(10,0), c(10,10), c(0,10), c(0,0))
p11<-rbind(c(1,1), c(2,1), c(2,2), c(1,2), c(1,1))
p12<-rbind(c(4,1), c(5,1), c(5,2), c(4,2), c(4,1))

mpol10<-st_multipolygon(list(list(p10,p11,p12)))
mpol11<-st_multipolygon(list(list(p11)))
mpol12<-st_multipolygon(list(list(p12)))

mal2<-st_sfc(list(mpol10,mpol11,mpol12))
mal<-st_sf(geom=mal2)
map_coloring(mal)
ggplot(mal, aes(fill = map_coloring(mal,TRUE))) + geom_sf() + theme_map()
```



Vzal jsem data za ČR a SR a R kód jsem si nechal vygenerovat. Skrz úsporu místa zde 22,6 MB zdrojový kód nevkládám, ale mohu říci, že mapa se vybarvila pěti barvami. My bychom si přáli ony slavné čtyři barvy. Teorém říká, že to jde, ale neříká, jak toho docílit.



4. Práce se surovými daty

Jsem surovomil, čím surovější data, tím milejší. Data jsem si stáhl z `https://osm-boundaries.com/`. Prvně jsem užil Overpass, `https://overpass-api.de/`, ale vedle hranic byly ve výsledku další údaje, neměl jsem dost sil na extrakci potřebného.

Pak mě napadlo, že by se hodilo znát sousedské vztahy. Stručně vysvětlím. Vzal jsem si všechny body hranic okresů a u nich jsem si uložil informaci, který okres je obsahuje. Pokud bod nemá souseda, je v takovém seznamu jeden okres. Ve většině případů jsou tam dva okresy, občas víc, jak se v jednom bodu setkává víc okresů. Z těchto informací, když je smrskneme na neopakující, lze získat sousedské vztahy. Tip: u Prahy bylo potřeba si dát pozor, že jsou obvody, ale i celá Praha. Tedy body na hranicích Prahy měly 3 údaje: Prahu-východ či Prahu-západ, Prahu celkovou, a pak některou z obvodů Praha 1–10. Další zvláštnost byla Brno, tam se nesmělo zapomenout, že se skládá z vnějšího a vnitřního polygonu.

Další úprava dat spočívala v tom, že byl požadavek mít společnou hranici pro Bratislavu I–V a Košice I–IV. Využil jsem předchozího bádání a vnější body o jednom údaji tvořily základ nové hranice. Plus se užily spojovací body mezi okresy. Byly to krajní body na těch úsecích. Tato finta mě potěšila, že jsem na to přišel. Jak by se to mělo řešit mimo surová data, ať už v R, Pythonu či jinými nástroji: nedokáži vám prozradit.

Pokud jste „rekurziholik“ (s úsměvem: srovnejte alkohol a alkoholik, nebo excesivní rekurzik či snad dokonce rekurzák) jako má maličkost, tím se vždy práce začíná. Zde je ukázka kódu, který sice teoreticky funguje, ale časově nám začne kolabovat kolem 60 okresů.

```
def maldfs(position):
  global visited, colors
  if position==166:
```

```
    print(colors)
    exit()
  col=["red","green","blue","yellow"]
  for okres in okresy[position]["sousedi"]:
    if colors[okres] in col:
      col.remove(colors[okres])
  for c in col:
    visited[position]=True;  colors[position]=c
    maldfs(position+1)
    visited[position]=False; colors[position]="none"

visited={}; colors={}
for k in range(165+1):
  visited[k]=False
  colors[k]="none"
maldfs(0)
```

Struktura vstupních dat, přesněji vyjádření sousedských vztahů, vypadala velmi jednoduše:

```
okresy={
  # [...]
  114: {'poradi':114, 'name':'okres Nitra', 'sousedi':[90, 96, 110, 119, 124, 126, 145]},
  115: {'poradi':115, 'name':'okres Veľký Krtíš', 'sousedi':[100, 132, 133, 134, 145]},
  # [...]
  }
```

Měl jsem radost, ale na 166 okresů to byl nedostatečný přístup.

5. Optimalizace na pomoc

Během jednoho večera jsem narazil na tuto stránku `https://www.reddit.com/r/math/comments/7uhtci/four_colour_theorem_is_there_an_existing/`. Zahlédl jsem tyto dva řádky od uživatele you-get-an-upvote na redditu a bylo mi hned jasné, jak to zpytlíkovat. Někdy stačí málo. Jen dopisuji čárky do druhého řádku, ty tam chyběly a souseda jsem upravil na k z j.

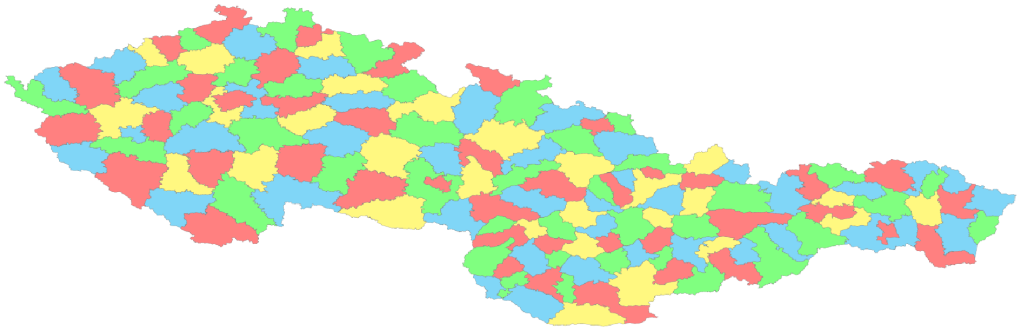
```
red(i)+green(i)+blue(i)+yellow(i) = 1
red(i)+red(k)<=1, green(i)+green(k)<=1, blue(i)+blue(k)<=1, yellow(i)+yellow(k)<=1
```

Mějme SAT model s proměnnými x_{ij}, kdy za i máme zvažované okresy, za j barvu 1–4. Dále chtějme tuto proměnnou binárního typu. Sečteme-li proměnné za okres i, musí být součet roven 1, tedy barva ze čtyř je použita jen jedna a právě jedna. To říká ten první řádek. Druhý řádek nám říká, že libovolní dva sousedi i a k mohou užít konkrétní barvu maximálně jednou. Záměrně jsem si model nepřevedl na minimalizační či maximalizační, chtěl jsem, řekněme, barvy rovnoměrně či náhodně jako překvapení.

Zde je zdrojový kód pro Python, který generuje soubor pro program Picat, `https://picat-lang.org/`, který se v závěru spustí. Struktura vstupních dat byla již představena.

```
import os
# Načtení vstupních dat.
```

```
kam=open("forpicat.pi","w")
kam.write("import sat.\n")
kam.write("main =>\n")
kam.write("Sol=[")

# Definování proměnných.
data=[]
for o in okresy:
  for b in [1,2,3,4]:
    data.append("Xo"+str(o)+"b"+str(b))
kam.write(",".join(data))
kam.write("],\n")
kam.write("Sol :: 0..1,\n")
kam.write("\n")

# Seznam okresů, které se neužijí, Praha, Bratislava a~Košice.
forbidden=[0,1,2,3,4,5,6,7,8,9,  112,141,144,147,  116,117,118]
# Jen jedna barva na okres.
for o in okresy:
  data=[]
  for b in [1,2,3,4]:
    data.append("Xo"+str(o)+"b"+str(b))
  if o not in forbidden:
    kam.write("+".join(data)+" #= 1,\n")
kam.write("\n")

# Pomocné proměnné, které se v závěru vypíší.
vypis=[]
for o in okresy:
  data=[]
  for b in [1,2,3,4]:
    data.append(str(b)+"*Xo"+str(o)+"b"+str(b))
  kam.write("Xo"+str(o)+" #= "+"+".join(data)+",\n")
  vypis.append("Xo"+str(o))
kam.write("\n")

# Bonusy, u neužitých obvodů/okresů dej barvu "none".
for kkk in forbidden: # Praha, Bratislava a~Košice.
  kam.write("Xo"+str(kkk)+" #= 0,\n")
kam.write("\n")

# Jádro, sousedské vztahy.
for o in okresy:
  for soused in okresy[o]["sousedi"]:
    for barva in [1,2,3,4]:
      kam.write("Xo"+str(o)+"b"+str(barva)+" + Xo"+str(soused)+"b"+str(barva)+" #<=1,\n")
kam.write("\n")

kam.write("Vypis=["+",".join(vypis)+"],\n")
kam.write("solve(Vypis),\n")
kam.write("writeln(Vypis).\n")
kam.close()

os.system("picat forpicat.pi")
```



6. Úvaha místo Závěru

Zaujala mě tato webová stránka, `https://carto.com/blog/sql-graph-coloring`, byť pravidelně bych takto v SQL nechtěl programovat.

Bylo by zajímavé zjistit si velikost jednotlivých oblastí a nějak to zahrnout do SAT→ILP modelu. Například aby čtyři barvy byly co nejvíc rovnoměrně či naopak co nejvíc rozházené. Nikdy jsem to nezkusil a těžko se představují grafické výstupy, které by z toho mohly vzniknout. Třeba nápad pro někoho ze čtenářů!

Reference

[1] R. M. R. Lewis: *A Guide to Graph Colouring: Algorithms and Applications,* 2nd edition, Springer Cham, 2021. `https://rhydlewis.eu/gcol/`

[2] Georges Gonthier: *A computer-checked proof of the Four Color Theorem.* `https://inria.hal.science/hal-04034866/document`

[3] K. Appel, W. Haken: *Every Planar Map is Four Colorable, Part I: Discharging,* Illinois Journal of Mathematics 21 (1977), 429–90. `https://projecteuclid.org/journals/illinois-journal-of-mathematics/volume-21/issue-3/Every-planar-map-is-four-colorable-Part-I-Discharging/10.1215/ijm/1256049011.full`

[4] K. Appel, W. Haken: *Every Planar Map is Four Colorable, Part II: Reducibility,* Illinois Journal of Mathematics 21 (1977), 491–567. `https://projecteuclid.org/journals/illinois-journal-of-mathematics/volume-21/issue-3/Every-planar-map-is-four-colorable-Part-II-Reducibility/10.1215/ijm/1256049012.full`

Kontaktní adresa

Ing. Pavel Stříž, Ph.D., Nakladatelství Martin Stříž, U Škol 940, Bučovice, okres Vyškov, 685 01, Česká republika,

E-mailová adresa: `pavel@striz.cz`